BAB II
LANDASAN TEORI

2.1 Citra

Definisi citra menurut Kamus Webster adalah "suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda". Sebagai contoh, foto seseorang mewakili entitas orang tersebut di depan kamera, Foto sinar X *thorax* mewakili keadaan bagian dalam tubuh seseorang, dan data dalam suatu file GIF mewakili apa yang digambarkannya (Firdausy et al, 2005).

Citra dapat digolongkan menjadi citra tampak dan citra tak tampak. Banyak contoh citra tampak, antara lain : foto keluarga, lukisan Picasso. Sedangkan contoh citra tak tampak antara lain data gambar dalam file (citra *digital*), dan citra yang direpresentasikan menjadi fungsi matematis.

Diantara jenis-jenis citra tersebut, hanya citra *digital* yang dapat diolah menggunakan komputer. Jenis citra lain jika hendak diolah dengan komputer, harus diubah dulu menjadi citra *digital*, misalnya foto dipindai (*scan*) dengan *scanner*, informasi densitas dan

komposisi bagian dalam tubuh manusia ditangkap dengan bantan pesawat sinar X.

Agar dapat diolah dengan komputer secara *digital*, maka suatu citra harus direpresentasikan secara numerik dengan nilai-nilai diskrit. Representasi citra dari dari fungsi kontinyu menjadi nilai-nilai diskrit disebut **digitalisasi** (Munir,2004).

Pada umunya citra digital berbentuk matriks, yang memiliki dimensi ukuran yang dinyatakan sebagai (n x m) atau tinggi kali lebar atau lebar kali panjang. Citra digital yang berukuran (n x m) biasanya dinyatakan dengan matriks yang dinotasikan :

m = kolom, n = baris

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & ..... & f(0,m) \\ f(1,0) & f(1,1) & ..... & f(1,m) \\ f(n-1,0) & f(n-1,1) & ..... & f(n-1,m-1) \end{bmatrix}$$

Indeks baris(n) dan indeks kolom(m) menyatakan suatu koordinat titik pada citra, sedangkan f(x,y) merupakan intensitas (derajat keabuan) pada titik (x,y).

Secara matematis fungsi intensitas cahaya pada bidang 2 Dimensi disimbolkan dengan f(x,y), yang dalam hal ini:

(x,y) : koordinat pada bidang 2 dimensi

f(x,y) : intensitas cahaya pada titik (x,y)

Karena cahaya merupakan bentuk energi, maka intensitas cahaya bernilai antara 0 sampai tidak berhingga atau bisa ditulis dengan $0 \leq f(x,y) < \infty$.

Nilai f(x,y) sebenarnya adalah hasil kali dari (Munir,2004):

1. i(x,y) = jumlah cahaya yang berasal dari sumbernya (*illumination*), nilainya antara 0 sampai tidak berhingga, dan
2. r(x,y) = derajat kemampuan objek memantulkan cahaya (*reflection*), nilainya antara 0 dan 1.

Nilai i(x,y) ditentukan oleh sumber cahaya, sedangkan r(x,y) ditentukan oleh karakteristik obyek di dalam gambar. Nilai r(x,y) = 0 mengindikasikan penerapan total, sedangkan r(x,y) = 1 menyatakan pemantulan total. Jika permukaan mempunyai derajat pemantulan nol, maka fungsi intensitas cahaya, f(x,y), juga nol. Sebaliknya, jika permukaan mempunyai derajat pemantulan 1, maka fungsi intensitas cahaya sama dengan iluminasi yang diterima oleh permukaan tersebut. Intensitas f dari gambar hitam putih pada titik (x,y) disebut **derajat keabuan** (*gray level*), yang dalam hal ini derajat keabuannya bergerak dari hitam ke putih,

sedangkan citranya disebut **citra hitam-putih atau citra skala keabuan** (*grayscale image*).

Derajat keabuan memiliki rentang nilai dari I_{min} sampai I_{max}, atau biasa ditulis dengan $I_{min} < f < I_{max}$. Selang (I_{min} , I_{max}) disebut **skala keabuan.**

Sebagai contoh, citra hitam-putih dengan 256 level, artinya mempunyai skala keabuan dari 0 sampai 255 atau [0,255], yang menyatakan nilai intensitas 0 adalah hitam, intensitas 255 menyatakan putih, sedangkan nilai intensitas 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih.

2.2 Pengolahan Citra

Pada awalnya pengolahan citra dilakukan untuk memperbaiki kulitas citra, seperti menambah kontras pada citra, mengurangi derau atau *noise*. Namun seiring dengan berkembangnya dunia komputasi yang ditandai dengan semakin meningkatnya kapasitas dan kecepatan proses komputer, serta munculnya ilmu-ilmu komputasi memungkinkan manusia untuk dapat mengambil informasi dari citra. Oleh karena itu *image processing* digunakan untuk kebutuhan yang lebih luas, antara lain pengenalan sidik jari maupun pengenalan wajah manusia, bahkan

untuk menyimpan pesan pada suatu citra. Bahkan dalam bidang kedokteran, dengan bantuan sinar-X aplikasi pengolahan citra digunakan untuk melihat bagian dalam tubuh manusia.

Suatu citra *digital* melalui pengolahan citra *digital* (*digital image processing*) akan menghasilkan citra *digital* yang baru, termasuk di dalamnya adalah perbaikan citra (*image restoration*) dan peningkatan kualitas citra (*image enhancement*). Sedangkan analisis citra digital (*digital image analysis*) akan menghasilkan suatu keputusan atau suatu data; termasuk di dalamnya adalah pengenalan pola (*pattern recognition*). (Firdausy et al, 2005)

Gambar 2.1 Urutan pengolahan citra digital (Firdausy et al, 2005)

2.3 Operasi-operasi Dasar Pengolahan Citra Digital

2.3.1 Operasi Komputasi

Operasi-operasi yang dilakukan pada pengolahan citra dapat dikelompokan dalam empat level komputasi, yaitu level titik, level lokal, level global, dan level objek (Munir,2004).

2.3.1.1 Level Titik

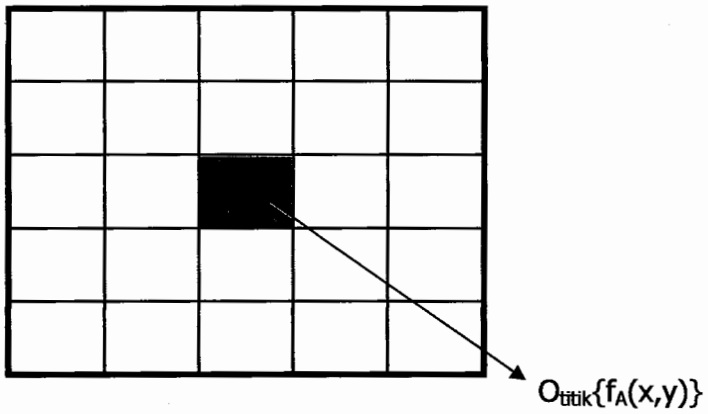
Operasi pada level titik hanya dilakukan pada piksel tunggal di dalam citra. Operasi titik dikenal juga dengan nama operasi *pointwise*. Operasi ini terdiri dari pengaksesan piksel pada lokasi yang diberikan, memodifikasi dengan operasi linear atau nonolinear, dan menempatkan nilai piksel baru pada lokasi yang bersesuaian di dalam citra baru. Operasi ini diulangi untuk keseluruhan piksel di dalam citra.

Secara sistematis, operasi pada level titik dinyatakan sebagai:

$$f_B(x,y) = O_{titik}\{f_A(x,y)\} \qquad (2.1)$$

Keterangan :

f_A dan f_B masing-masing adalah citra masukan dan citra keluaran O_{titik} dapat berupa operasi linear atau operasi nonlinear.

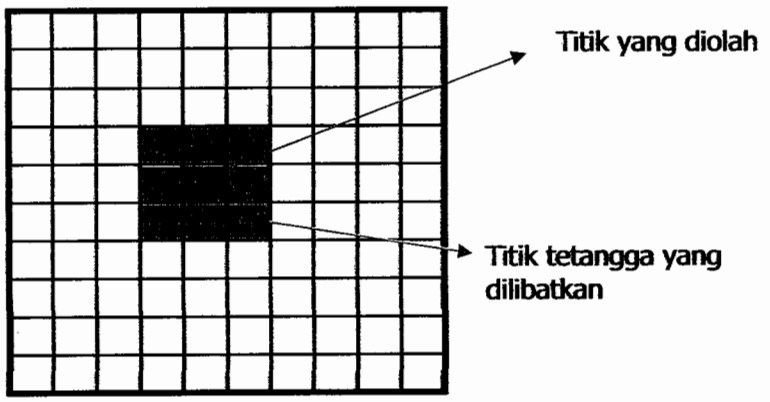


Gambar 2.2 Operasi level titik
(Dwiandiyanta, 2005)

Operasi pada level titik dapat dibagi menjadi tiga macam, yaitu berdasarkan intensitas, geometri, atau gabungan keduanya. Sebagai contoh adalah operasi pengambangan, operasi negatif, pemotongan (*clipping*), pencerahan citra. Contoh operasi titik berdasarkan geometri seperti rotasi, translasi (pergeseran), dilatasi (penskalaan). Sedangkan operasi titik gabungan seperti *image morphing*, yaitu perubahan bentuk objek beserta intensitasnya.

2.3.1.2 Level Lokal

Operasi pada level lokal menghasilkan citra keluaran yang intensitas suatu piksel bergantung pada intensitas piksel-piksel tetangganya. Operasi ini disebut juga dengan operasi konvolusi. Operasi level lokal selain dilihat dari nilai piksel itu sendiri, nilai-nilai piksel tetangganya juga turut mempengaruhi hasil operasi, karena turut diperhitungkan. Pengertian tetangga di sini sangat dinamis dalam hal jarak, artinya radius tetangga dalam satuan piksel sudah ditentukan lebih dahulu dalam algoritma pengolahan citra. Contoh operasi level lokal adalah operasi konvolusi untuk mendeteksi tepi atau pelembutan citra (*image smoothing*) atau dengan kata lain pengaburan citra untuk memperlemah keberadaan *noise*.

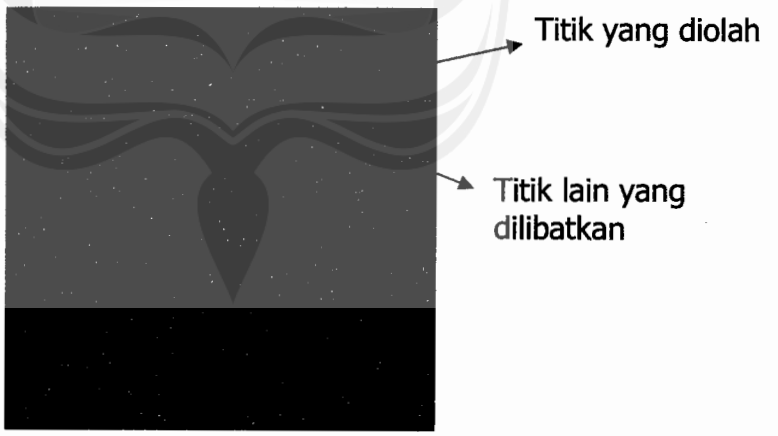


Gambar 2.3 Operasi Level Lokal
(Dwiandiyanta, 2005)

2.3.1.3 Level Global

Operasi pada level global menghasilkan citra keluaran yang intensitas suatu piksel bergantung pada intensitas keseluruhan piksel. Pada operasi level global seluruh bagian citra diperhitungkan sehingga hasilnya akan tergantung pada keadaan citra keseluruhan. Hasil dari operasi level global, untuk citra yang sama dengan kualitas yang berbeda (misalnya citra yang satu lebih terang dari citra lainnya secara keseluruhan) akan berbeda. Contoh operasi level global adalah operasi penyetaraan histogram untuk meningkatkan kualitas gambar. Operasi ini digunakan

difungsikan untuk meningkatkan kualitas gambar atau memperbaiki kulitas citra, dan juga dapat memperoleh lembah dari 2 puncak banyakanya intensitas piksel. Dalam operasi pembuatan histogram, seluruh piksel dalam sebuah citra harus dibaca dan didata dahulu, kemudian piksel-piksel dengan nilai intensitas yang sama akan dikelompokkan dan jumlahnya dihitung, baru kemudian dibuat. Jadi bentuk histogram dipengaruhi oleh keadaan citra secara keseluruhan atau global, bukan hanya oleh sekelompok piksel saja.



Gambar 2.4 Operasi Level Global
(Dwiandiyanta, 2005)

2.3.1.4 Level Objek

Operasi level objek hanya dapat dilakukan pada objek tertentu di dalam citra. Tujuan dari operasi pada level objek adalah untuk mengenali objek, misalnya dengan menghitung rata-rata intensitas, ukuran, bentuk, dan karakteristik lain dari objek. Sebenarnya banyak karakteristik suatu obyek yang harus dihitung, misalnya ketika ingin mendeteksi kehadiran faktor-faktor penyebab cacat pada produk melalui analisis bentuknya, yaitu mungkin harus menghitung ukuran, faktor bentuk, posisi dari obyek dan sebagainya. Pembahasan pengenai lebih lanjut akan dibahas pada tahap analisa program.

2.3.2 Operasi Aritmatika

Karena citra *digital* merupakan matriks, maka operasi aritmatika matriks juga berlaku pada citra.

2.3.2.1 Penjumlahan/pengurangan 2 citra

Persamaannya yang digunakan dalam operasi ini adalah :

Penjumlahan: $C(x,y)=A(x,y)+B(x,y)$ (2.2)

Pengurangan: $C(x,y)=A(x,y)-B(x,y)$ (2.3)

Dengan C(x,y) adalah citra baru yang setiap pikselnya adalah jumlah atau selisih dari intensitas tiap piksel pada A dan B.

Jika hasil penjumlahan lebih besar dari 255, maka intensitas dapat dibulatkan menjadi 255. Namun apabila hasil pengurangan lebih kecil dari 0 atau negatif, maka akan diperlukan proses *clipping*.

Penjumlah citra sering digunakan untuk penggabungan dua buah citra dan *watermarking* tampak (*visible watermarking*), sedangkan pengurangan citra sering digunakan untuk untuk mendeteksi perubahan objek dalam selang waktu tertentu.

Gambar 2.5 Contoh operasi penjumlahan citra (Dwiandiyanta, 2005)

2.3.2.2 Perkalian 2 buah citra

Persamaan yang digunakan untuk operasi ini adalah :

$$C(x,y)=A(x,y)B(x,y) \quad (2.4)$$

Perkalian citra sering digunakan untuk mengoreksi ketidaklinearan sensor dengan mengalikan matriks citra dengan matriks koreksi. Jadi, dalam hal ini A adalah citra, sedangkan B adalah matriks koreksi. Hasil operasi mungkin bernilai riil, karena itu semua nilai dibulatkan ke nilai bulat terdekat. Nilai maksimum operasi ini 255.

2.3.2.3 Penjumlahan/pengurangan citra dengan skalar

Persamaan yang digunakan untuk operasi ini adalah :

$$B(x,y) = A(x,y) \pm c \quad (2.5)$$

Penjumlahan/pengurangan citra A dengan skalar c adalah menambah setiap piksel di dalam citra dengan sebuah skalar c, dan menghasilkan citra baru B yang intensitasnya lebih terang/ gelap dibandingkan dengan citra A.

Hasil penjumlahan atau pengurangan citra dengan skalar mungkin menghasilkan nilai dengan intensitas negatif atau lebih dari 255, sehingga diperlukan proses *clipping*

Citra asli | Citra dengan *brightness* ditambah 50 | Citra dengan *brightness* dikurangi 50

Gambar 2.6 Contoh Operasi penjumlahan/pengurangan citra dengan skalar (Dwiandiyanta, 2005)

2.3.2.4 Perkalian/pembagian citra dengan skalar

Persamaan yang digunakan untuk operasi ini adalah :

Perkalian : $B(x,y)=A(x,y).c$ (2.6)

Pembagian : $B(x,y)=A(x,y)/c$ (2.7)

Perkalian citra A dengan c akan menghasilkan citra baru B dengan intensitas yang lebih terang dibandingkan dengan citra A. Kenaikan intensitas tersebut akan sebanding dengan c.

Pembagian citra A dengan c akan menghasilkan citra baru B dengan intensitas yang lebih gelap dibandingkan dengan citra A. Penurunan intensitas akan berbanding terbalik dengan c.

Kedua operasi tersebut digunakan untuk normalisasi kecerahan pada citra.

Citra asli | Citra dengan *brightness* dikalikan 1,5 | Citra dengan *brightness* dibagi 1,5

Gambar 2.7 Contoh Operasi perkalian/pembagian citra dengan skalar (Dwiandiyanta, 2005)

2.3.3 Operasi Boolean

Operasi Boolean yang dapat diterapkan pada citra, antara lain :

Operasi Boolean **and**

$$C(x,y)=A(x,y) \wedge B(x,y) \quad (2.8)$$

Operasi Boolean **or**

$$C(x,y)=A(x,y) \vee B(x,y) \quad (2.9)$$

Operasi Boolean **not**

$$C(x,y)=\sim A(x,y \quad (2.10)$$

Operasi boolean pada citra mempunyai terapan yang penting pada pemrosesan morfologi citra biner. Pada citra biner, operasi **not** dapat digunakan untuk menentukan komplemen dari citra.

Citra Asli Citra Hasil Operasi NOT

Gambar 2.8 Contoh Operasi ***not***
(Dwiandiyanta, 2005)

2.3.4 Operasi Geometri

Pada operasi geometri, koordinat *pixel* berubah akibat transformasi, sedangkan intensitasnya tetap. Hal ini berbeda dengan operasi Aritmatika yang mana koordinat *pixel* tetap sedangkan intensitasnya berubah.

2.3.4.1 Translasi

Operasi Translasi dilakukan berdasar rumus :

$$x' = x+m \quad (2.11)$$

$$y' = y+m \quad (2.12)$$

m adalah besarnya pergeseran dalam arah x, sedangkan n adalah besarnya pergeseran dalam arah y. Jika citra semula adalah A, dan citra hasil translasi adalah B, maka

translasi yang dapat dilakukan adalah sebagai berikut:

$$B(x,y) = A(x+m,y+n) \quad (2.13)$$

2.3.4.2 Rotasi

Operasi Rotasi dilakukan dengan persamaan:

$$x' = x \cos(\theta) - y \sin(\theta) \quad (2.14)$$

$$y' = x \sin(\theta) + y \cos(\theta) \quad (2.15)$$

Dalam hal ini, θ adalah sudut rotasi berlawanan dengan arah jarum jam. Jika citra semula adalah A, dan citra hasil rotasi adalah B, maka rotasi citra dari A ke B :

$$B(x,y) = A(x \cos(\theta) - y \sin(\theta), x \sin(\theta) + y \cos(\theta)) \quad (2.16)$$

2.3.4.3 Penskalaan Citra

Penskalan citra disebut juga *image zooming*, yaitu pengubahan ukuran citra (memperbesar/*zoom out* atau memperkecil/*zoom in*). Rumus penskalaan citra:

$$x' = s_x.x \quad (2.17)$$

$$y' = s_y.y \quad (2.18)$$

s_x dan s_y adalah faktor penyekalaan, masing-masing dalam arah x dan y. Jika citra semula adalah A dan citra hasil penyekalaan adalah B, maka penyekalaan citra dinyatan sebagai:

$$B(x',y')=B(s_x.x,s_y.y)=A(x,y) \qquad (2.19)$$

2.3.4.4 Flipping

Flipping adalah operasi geometri yang sama dengan pencerminan (*image reflection*). Ada 2 macam *flipping*, yaitu :

- Horisontal

  Adalah pencerminan terhadap sumbu Y.

  Rumus : $B(x,y) = A(N-x,y)$ (2.20)

- Vertikal

  Adalah pencerminan terhadap sumbu X.

  Rumus : $B(x,y) = A(x,M-y)$ (2.21)

Gambar 2.9 Contoh Proses Flipping (Dwiandiyanta, 2005)

Selain kedua proses flipping tersebut, terdapat juga proses pencerminan terhadap titik asal. Persamaan yang digunakan adalah

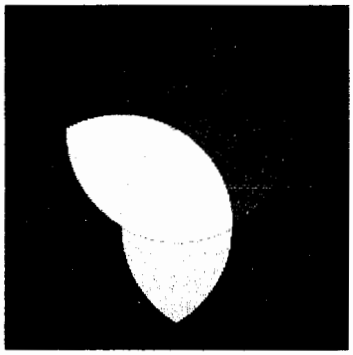
$$B(x,y)=A(N-x, M-y) \quad (2.22)$$

Dengan N adalah jumlah kolom citra, dan M adalah jumlah baris citra.

2.4 RGB & HSV

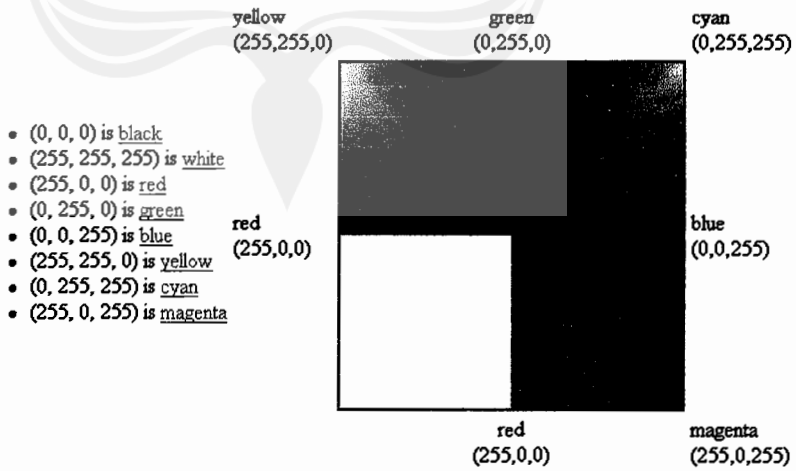
2.4.1 RGB

RGB adalah suatu pemodelan warna yang terdiri dari warna merah, biru, dan hijau yang dapat dikombinasikan dengan berbagai cara untuk merepresentasikan suatu warna. Berbeda dengan warna dasar dari suatu seni lukis(*art*), yaitu merah, biru, dan kuning, RGB merupakan kepanjangan dari *Red, Green, Blue* yang merupakan warna dasar dari sinar katoda. Dalam pemodelan warna RGB, bukan berarti suatu citra hanya dapat tersusun atas 3 warna tersebut, melainkan dapat tersusun atas campuran ketiga warna tersebut, sehingga bersifat relatif sesuai representasi dari mata manusia itu sendiri (*wikipedia.org*).



2.10 Model warna RGB (wikipedia.org)

Dalam citra bitmap 24 bit atau yang sering disebut dengan citra *True Color*, RGB dimodelkan ke dalam 3 buah bilangan bulat (*integer*) antara 0-255, di mana masing-masing bilangan tersebut mewakili intensitas dari warna merah (*red*), hijau (*green*), dan biru (*blue*). Di bawah ini merupakan contoh gambar yang dihasilkan dari sinar katoda.



Gambar 2.11 Model RGB pada citra True Color (wikipedia.org)

2.4.2 HSV

HSV adalah singkatan dari *Hue, Saturation, Value* atau disebut juga dengan sebutan HSB (*Hue, Saturation, Brightness*) yang merupakan komponen dari model warna HSV itu sendiri. (*wikipedia.org*)

2.4.2.1 Hue

Hue merupakan representasi dari suatu warna. Biasanya bernilai antara 0-360. Nilai 0 menunjukkan warna merah, nilai 240 menunjukkan warna biru.

Gambar 2.12 Skala nilai Hue (*wikipedia.org*)

2.4.2.2 Saturation

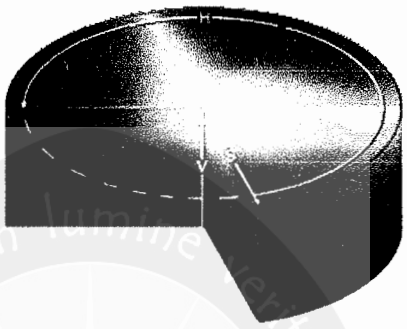
Merupakan tingkat intensitas suatu warna. Bernilai antara 0-100%. Nilai 0 menunjukkan warna diantara putih, hitam, dan abu-abu atau bahkan tidak berwarna. Nilai 100 menunjukkan warna dengan intensitas paling tinggi.

2.4.2.3 Value

Menunjukkan tingkat kecerahan dari suatu warna. Bernilai antara 0-100%. Nilai

0 menunjukkan warna hitam. Nilai 100 menunjukkan kedalaman warna tertinggi, tergantung dari nilai *Saturation*.

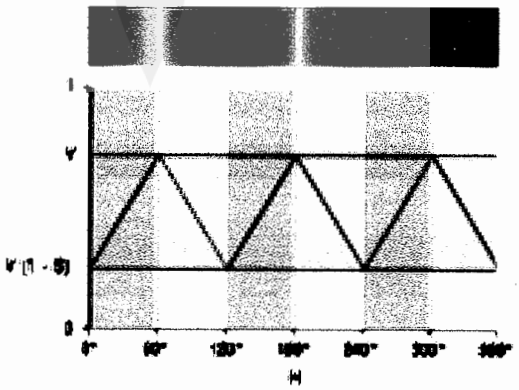
Hubungan antara *Hue, Saturation, dan Value* ditunjukkan dengan gambar berikut :



Gambar 2.13 Pemodelan HSV (wikipedia.org)

2.4.3 Transformasi RGB ke HSV dan HSV ke RGB

Hubungan antara HSV dan RGB ditunjukkan oleh gambar berikut :



Gambar 2.14 Hubungan HSV dan RGB (wikipedia.org)

2.4.3.1 RGB ke HSV

MIN : Nilai minimum dari RGB

MAX : Nilai maximum dari RGB

$$H = \begin{cases} \text{undefined}, & \text{if } MAX = MIN \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{if } MAX = R \text{ and } G \geq B \\ 60° \times \frac{G-B}{MAX-MIN} + 360°, & \text{if } MAX = R \text{ and } G < B \\ 60° \times \frac{B-R}{MAX-MIN} + 120°, & \text{if } MAX = G \\ 60° \times \frac{R-G}{MAX-MIN} + 240°, & \text{if } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{otherwise} \end{cases}$$

$$V = MAX$$

2.4.3.2 HSV ke RGB

$$H_i = \left\lfloor \frac{H}{60} \right\rfloor \mod 6$$

$$f = \frac{H}{60} - H_i$$

$$p = V(1 - S)$$

$$q = V(1 - fS)$$

$$t = V(1 - (1 - f)S)$$

if $H_i = 0 \rightarrow R = V, G = t, B = p$

if $H_i = 1 \rightarrow R = q, G = V, B = p$

if $H_i = 2 \rightarrow R = p, G = V, B = t$

if $H_i = 3 \rightarrow R = p, G = q, B = V$

if $H_i = 4 \rightarrow R = t, G = p, B = V$

if $H_i = 5 \rightarrow R = V, G = p, B = q$

2.5 Delphi

Ide munculnya Delphi sebenarnya berasal dari bahasa pemrograman yang cukup terkenal, yaitu Pascal. Bahasa Pascal sendiri telah diciptakan pada tahun 1971 oleh ilmuwan dari Swiss, yaitu Niklaus Wirth. Nama Pascal diambil dari ahli matematika dan filsafat dari Prancis, yaitu Blaise Pascal (1623-1662).

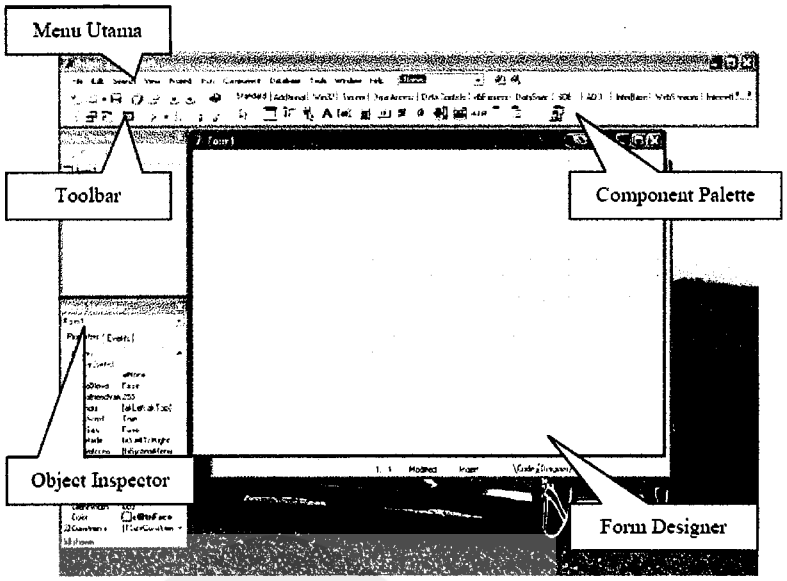
Sejak saat itu, muncul beberapa versi Pascal di antaranya Turbo Pascal yang dirilis oleh Borland International pada tahun 1983. Turbo Pascal yang muncul pertama kali hanya dapat dijalankan di sistem operasi DOS. Namun dalam perkembangan selanjutnya, Borland International juga merilis Turbo Pascal yang bisa berjalan di Windows 3.x, yaitu Turbo Pascal for Windows.

Umumnya Delphi lebih banyak digunakan untuk pengembangan aplikasi *desktop* dan *enterprise* berbasis *database*, tapi sebagai perangkat pengembangan yang bersifat *general-purpose* Delphi juga mampu digunakan dalam berbagai jenis proyek pengembangan *software*. Delphi juga yang dikenal sebagai salah satu yang membawa istilah *RAD* tool, kepanjangan dari *Rapid Application Development*, saat dirilis tahun 1995 untuk windows 16-bit. Delphi 2 dirilis setahun kemudian yang

mendukung lingkungan windows 32-bit, dan versi C++, C++Builder dirilis beberapa tahun kemudian. Pada tahun 2001 sebuah versi linux yang dikenal sebagai Kylix tersedia. Dengan satu rilis baru setiap tahunnya, pada tahun 2002 dukungan untuk Linux (melalui Kylix dan CLX component library) ditambahkan dan tahun 2003 .NET mulai didukung dengan munculnya Delphi.Net (Delphi 8).

2.5.1 IDE pada Delphi

Delphi dapat digunakan untuk membuat berbagai jenis program, diantaranya aplikasi, ActiveX, WebService, dan lain-lain. Delphi menyediakan *Integrated Development Environtment* (IDE) untuk memudahkan perancangan tampilan untuk pemakai (antarmuka pemakai) secara visual dan penulisan kode. IDE Delphi menyediakan berbagai jendela yang akan sering digunakan oleh *programmer* maupun *developer* dalam pengembangan aplikasi, antara lain menu utama, *SpeedBar*, Jendela Form, *Object Inspector*, dan Komponen *Palette*.



Gambar 2.10 IDE Delphi
(Pranata, 2002)

2.5.2 *Visual Component Library* (VCL)

Visual Component Library merupakan pustaka yang berisi komponen visual yang biasa digunakan untuk membangun aplikasi dengan menggunakan Delphi. Pada Kylix, Borland juga merilis CLX yang merupakan VCL versi cross-platform, yaitu pustaka yang membungkus Windows API sekaligus Linux API, sehingga CLX dapat dijalankan pada platform Windows maupun Linux. Namun, tidak semua komponen pada VCL dapat digunakan pada aplikasi CLX, karena tidak semua komponen pada VCL bersifat *multi-platform*.

Komponen-komponen VCL diletakkan pada bagian *Component Palette* dengan dibagi menjadi beberapa *tab.* Masing-masing komponen mempunyai tiga atribut, yaitu properti, kejadian, dan metode. Oleh karena itu model pemrograman pada Delphi sering disebut juga model pemrograman PME (*Properties Method Event*) (Pranata, 2003).

Tidak semua komponen pada *Component Palette* adalah VCL, tetapi ada beberapa komponen yang berupa *ActiveX*. Cara penggunaan control ActiveX tidak banyak berbeda dengan VCL. Namun, terdapat dua perbedaan mendasar antara control ActiveX dengan VCL. Perbedaan pertama, control ActiveX sudah menjadi standar komponen pada lingkungan Windows. Dengan demikian sebagian besar piranti pengembangan Windows, mulai dari Visual Basic, Visual C++, C++Builder, sampai Delphi bisa menggunakan control ActiveX. Hal ini berbeda dengan komponen VCL yang hanya bisa digunakan pada Delphi dan C++Builder. Perbedaan kedua, aplikasi yang menggunakan control ActiveX harus didistribusikan beserta file **.ocx** (selain file **.exe**). Apabila hanya mendistribusikan file **.exe**,

maka aplikasi tersebut tidak akan dapat dijalankan. Namun, apabila menggunakan komponen VCL, yang perlu didistribusikan hanya file **.exe** saja.

2.6 SQLite

SQLite merupakan suatu *embedded database*, dengan begitu semua data disimpan di dalam satu file tunggal, yang menjadikannya sangat mudah untuk dibagi, sangat mudah untuk dipindahkan dan secara umum, sangat mudah untuk di *manage*.

Database system semacam ini sangat bertentangan dengan *database* yang bekerja secara *client/server* seperti PostgreSQL ataupun MySQL. Arsitektur *database system* keduanya memungkinkan adanya bagian yang jelas antara *server database* dan *client database*. Dengan demikian, untuk kebutuhan data diakses ramai-ramai, pengguna tidak perlu mengatur *file sharing* secara *manual*, karena *database system*-nya sendiri telah mengizinkan. Dibandingkan dengan *embedded database*, *database* semacam ini jelas memiliki keunggulan seperti *multiuser native*, skema keamanan yang lebih fleksibel, dan umumnya, lebih mendukung sistem *enterprise*.

Untuk memilih *database system* itu sendiri, seorang pemrogram harus menganalisa terlebih dahulu apakah *database system* yang akan gunakan termasuk kategori yang embbeded atau yang bekerja secara client/server. Berikut ini adalah beberapa alasan untuk menggunakan *embedded database:*

- Sering memindah-mindahkan *database* dari satu tempat ke tempat lain.
- *Database* tidak digunakan untuk melayani kebutuhan *enterprise* (contoh: tidak perlu *clustering database*).
- Tidak membutuhkan banyak fitur *advanced* (*Trigger, stored procedure*, dan lain-lain).

Beberapa alasan SQLite sering digunakan :

- Secara umum sangat stabil.
- Merupakan *software* yang dilisensikan *public domain*.
- Sangat cepat.
- Mendukung banyak sekali standar SQL92.
- Mampu menampung data sampai 2 Tera.
- Tidak memerlukan banyak memori, yang sangat berguna untuk sistem yang juga *embedded*.

- Banyak bahasa pemrograman telah mendukung *database* ini.
- Mendukung ACID, bahkan pada kegagalan sistem atau *power*.
- Disimpan dalam satu *file* tunggal.
- Tabel dapat disimpan pada *file* terpisah, dan dapat di-*attach* ke *database* utama.

SQLite adalah *database system* yang dikembangkan oleh D. Richard HiPP dari HWACI Applied Software Research (drh@hwaci.com). SQlite dapat di-download di http://www.sqlite.org.

Pada SQLite 2.x, semua data akan dikonversi ke teks ASCII. Pada versi 3, apabila memungkinkan, maka data akan disimpan sebagai tipe yang didefiniskan oleh user. Penyimpanan ke format non-ASCII sangat penting untuk memungkinkan adanya tipe data BLOB (*Binary Large Object*). BLOB pada versi 2.x dimungkinkan apabila data telah diencode terlebih dahulu. Pada versi 2.8, karena semua data akan dikonversi ke teks ASCII, maka dengan demikian, data yang seharusnya disimpan ke dalam kolom satu bisa disimpan ke kolom lain. Hal ini mungkin terdengar sangat asing pada *database system* lain. Pada

SQLite versi 3, fitur ini masih dipertahankan, namun penanganannya telah diperbaiki. Pada saat data dimasukkan ke dalam kolom, SQLite 3 akan mencoba untuk melakukan konversi ke format yang telah didefinisikan. Apabila konversi tidak dimungkinkan, SQLite 3 tetap akan menyimpan data tersebut. Sebagai contoh, apabila data string ingin dimasukkan ke dalam kolom integer, maka SQLite akan memeriksa apakah data dapat dikonversi ke bilangan. Apabila dapat, maka konversi dilakukan dan data akan disimpan ke dalam kolom integer. Apabila tidak dapat dikonversi, maka data akan tetap disimpan sebagai string. Berikut ini adalah tipe-tipe data yang didukung:

1. NULL, nilai null.
2. INTEGER, menyimpan integer sampai integer 8 byte.
3. REAL, menyimpan nilai *floating point*/pecahan sampai 8 *byte* IEEE floating point.
4. TEXT, menyimpan teks.
5. BLOB, menyimpan data blob.

2.7 Photo Mosaic

Secara garis besar aplikasi **Photo Mosaic** ini terbagi atas berbagai tahap. Adapun tahap-tahap yang terdapat di dalam **Photo Mosaic** adalah sebagai berikut:

1. Apabila belum terdapat *library (database)*, maka *user* akan diminta untuk memasukkan citra mosaik ke dalam *library*, sekaligus memasukkan nama *database* untuk menyimpan citra mosaik tersebut. Di dalam penyimpanan citra mosaik, sistem akan me-*resize* ukuran citra menjadi 45 x 45, kemudian menghitung rata-rata HSV dari masing-masing citra mosaik. Penghitungan rata-rata HSV ini dilakukan dengan cara mengkonversikan nilai RGB tiap-tiap piksel pada citra mosaik menjadi HSV kemudian menghitung rata-rata keseluruhan HSV pada citra tersebut. Nilai HSV inilah yang nantinya akan disimpan pada kolom yang sudah disediakan dalam *database*.
2. Setelah terdapat *database* di dalam sistem, kemudian *user* akan diminta untuk memasukkan citra utama yang akan diolah, nama *database* yang akan digunakan dan jumlah citra mosaik yang akan digunakan untuk menyusun citra hasil.
3. Sistem akan me-*resize* ukuran citra utama sesuai dengan jumlah mosaik yang telah diinputkan oleh

user dengan tidak menghapus citra utama dengan ukuran aslinya. Kemudian sistem akan mengkonversikan nilai RGB tiap-tiap piksel pada citra utama menjadi nilai HSV. Lalu sistem akan mencari *nilai HSV dari citra mosaik di dalam* database yang paling mendekati nilai HSV dari masing-masing piksel pada citra utama.

4. Akan terbentuk suatu citra hasil yang tersusun atas citra-citra mosaik. Citra hasil ini akan berukuran 45 x jumlah mosaik yang dimasukkan oleh *user*. Sebagai contoh, apabila *user* memasukkan jumlah mosaik sebanyak 40 x 30, maka citra yang akan dihasilkan akan berukuran (45x40)x(45x30), yaitu 1800 x 1350.
5. Untuk menjaga detail pada citra utama, maka diperlukan operasi *crossfading* yang digunakan untuk menggabungkan antara citra utama dengan citra hasil. Sebelumnya sistem akan me*resize* ukuran citra utama menjadi berukuran sama dengan citra hasil. Dalam hal ini *user* akan diminta untuk memasukkan tingkat transparansi dari citra utama.
6. Hasil dari penggabungan inilah yang akan disimpan menjadi suatu citra baru dengan format bmp.